UNITED STATES PATENT APPLICATION

Selectable Control of Raster Image Processors

INVENTORS

Robert D. Christiansen
4586 N. Chapala
Boise, ID 83713

Scott Clouthier
8117 W. Arapaho Ct.
Boise, ID 83714

Amir Gaash
19 Frid Street
Rehovot, Israel

Ouri Poupko
41 Bialik Street
Nes-Ziona, Israel

Attorney Docket: 200207578-1

Selectable Control of Raster Image Processors

Background

Almost all digital printing and image-display technologies rely on a technique known as raster image processing to produce images on a passive or active display medium, such as paper or an electronic screen. In this technique, a hardware or software module, called a raster image processor (RIP or RIP engine), converts vector image data (a geometrical form of image data that facilitates image storage and manipulation) to raster data. The raster data, also known as bit-map data or hardware-ready bits, allows a printer or display to render the image as a pattern of dots or pixels.

In recent years, it has become common, particularly in professional printshops, to use two or more RIPs to rasterize some types of print jobs. For example, some jobs are so large that two or more RIPs are used to reduce the time spent rasterizing and thereby increase production rates. Also, some jobs include two or more types of vector image data, with each type requiring its own specialized form of RIP to convert it to raster data.

In both these cases, the multiple RIPs, generally referred to collectively as a RIP pipeline, are controlled by a RIP manager. The RIP manager, which can take the form of a hardware or software module, receives the print job, divides it into two or more parts, and then assigns each part to a particular RIP for rasterizing. Each of the RIPs then works independently to rasterize its part of the print job, notifying the RIP manager when done and thus allowing the RIP manager to reassign the RIP to another print job.

One problem that the present inventors recognized with conventional RIPs is that they have limited communications capabilities. In some situations, such as dealing with external clients (that is, other users of the RIP), these limited capabilities effectively reduce the processing speed of the RIPs.

Accordingly, the present inventors identified a need for RIPs with improved communications.

Brief Description of Drawings

Figure 1 is a block diagram of an exemplary printing system 100 corresponding to one or more embodiments of the present invention.

Figure 2 is flow chart of an exemplary method of operating one or more portions of printing system 100.

Detailed Description of Exemplary Embodiments

This description, which references and incorporates the above-identified figures and the appended claims, describes one or more specific embodiments of one or more inventions. These embodiments, offered not to limit but only to exemplify and teach the one or more inventions, are shown and described in sufficient detail to enable those skilled in the art to implement or practice the invention. Thus, where appropriate to avoid obscuring the invention, the description may omit certain information known to those of skill in the art.

Figure 1 shows an exemplary printing (or hardcopy imaging) system 10 which corresponds to one or more embodiments of the present invention. System 10 includes a digital front end (or print preprocessor) 12, a printer set 14, and an external client 16.

Digital front end 12 includes a RIP manager 122 (more generally a RIP control entity), a RIP pipeline 124, RIP 126, and processing resources 128. Generally, each of these components, with the exception of processing resources 128, comprises one or more modules, objects, or other groupings or encapsulations of underlying functionality implemented in a programming code, such as JAVA or C++, that is carried on an electronic, magnetic, or optical storage medium or carrier signal. These various components can be intercoupled by any form of communications link. For example, the exemplary embodiment assigns each of these components an IP (Internet Protocol) address and communicates via one or more socket-based network protocols. Some embodiments implement one or more of these components in hardware, for

example, as one or more application-specific integrated circuits and/or in firmware, for example, as one or more ROM modules.

More specifically, RIP manager 122, which is coupled to RIP pipeline 124 and RIP 126, includes a graphical user interface (GUI) module 1221 and a RIP database 1222. GUI module 1221 includes one or more graphical user interfaces which are displayed on a display device (not shown) and allow users to define or create one or more pipelines and associate one or more RIP engines with each pipeline, using a keyboard, mouse, or other input device (not shown) In particular, GUI module 1221 includes an input (or input region) 1221A, for example in the form of a check box or radio button, which allows users to enable or disable a RIP engine (such as RIP 126) from receiving and/or sending commands and data from entities other than RIP manager 122. Although not shown in the figure, the interface also includes one or more inputs (or input regions) for enabling automatic job acceptance and for setting or selecting automatic job-acceptance criteria, such as job size, job types, and job numbers, for separate RIP engines or RIP pipelines.

RIP database 1222 includes data identifying one or more RIPs (or RIP engines) and related status and address information. In the exemplary embodiment, database 1222 includes for each of one or more RIP engines: a corresponding network address and/or name; a corresponding installation directory path name and/or data; a corresponding current controlling RIP pipeline name or identifier; and corresponding location information for imposition, LUT, and ICC profile directories relative to the corresponding installation path.

RIP pipeline 124 can include two or more RIPs, of which a RIP 1241 is generally representative. (The RIPs in pipeline 124 can be real RIPs, virtual RIPs, or combinations of real and virtual RIPs.) RIP 1241, which is a conventional RIP in the exemplary embodiment, includes conventional RIP software modules and/or hardware (not shown) for rasterizing input vector image data--- that is, for converting the vector image data to bit-map (or hardware-ready) data. Examples of input vector image data include PostScript (PS) data,

Portable Document Format (PDF) data, and Printer Control Language (PCL) data.

RIP 126, which can be part of RIP pipeline 124 in some embodiments, includes a controller interface 1261 and a RIP engine 1262. Controller interface 1261 enables one of a set of two or more RIP control or management entities, such as RIP manager 122 and external client 16, to control or otherwise communicate with RIP engine 1262. In the exemplary embodiment, controller interface 1261 includes one or more network socket connections (ports, modules or objects), of which socket ports 1261A, 1261B, 1261C, and 1261D are generally representative.

Socket port 1261A is an input socket port for receiving packetized vector-image data from external client 16. Some embodiments include input socket ports that are devoted to particularly types of vector-image data. For example, one embodiment includes one input socket port for PS data and another for PDF data. Socket port 1261B is a control socket port for communicating packetized messages between the external client and the RIP module, and socket port 1261C is an output socket port for outputting packetized bit-map data from the RIP module to the external client (or other RIP control entity.) In some embodiments, socket port 1261C is part of an output plug-in module that provides the associated RIP module the capability to communicate its output data in packet form via the output port.

Other embodiments may use other forms of interface mechanisms or protocols between controller interface 1261 and external client 16. For example, some embodiments use a Harlequin SOAR (Scalable Open Architecture RIP) application programming interface (API), Adobe CPSI (Configurable PostScript Interpreter) API mechanism, or a hot-folder-based mechanism. The hot-folder-based mechanism may use a set of folders and/or directories (for example, input, error, success, and output folders) as intermediate storage for messages, with multi-controller interface and RIPs monitoring the folders for new messages. Some embodiments base the controller interface on a Common Object Request Broker Architecture (CORBA). Examples of this form of architecture include

System Object Model (SOM), Distributed SOM (DSOM), Component Object Model (COM), and Distributed COM (DCOM). Some embodiments use a Remote Procedure Call (RPC) architecture or protocol.

Socket port 1261D functions as an intermediary between socket ports 1261A-1261C and RIP engine 1262. In other words, socket port 1261D translates and/or reformats commands and data to and/or from the commands and data native to RIP engine 1262.

RIP engine 1262 provides conventional rasterizing of one or more types of vector-image data, such as PS data or PDF data. In some embodiments, RIP engine 1262 includes a two or more specialized RIP engines and thus constitutes a RIP pipeline.

Processing resource 128 comprises one or more processing entities 1281 and memory (or data storage) devices 1282 for implementing the functionality of RIP manager 122, RIP pipeline 124, and RIP 126. In the exemplary embodiment, processing entities 1281 include one or more integrated central processing units, and storage devices 1282 include one or more volatile or non-volatile memory devices. Although shown as a centralized arrangement in this exemplary embodiment, some embodiments distribute one or more of these resources across a network.

In addition to digital front end 12, system 10 includes printer set 14 and external client 16. Printer set 14 includes a set of one or more printing or hard-copy imaging devices, of which printers 142, 144, and 146 are generally representative. Exemplary printers include: HP Indigo 1000, HP Indigo 3000, HP Indigo w3200, HP LaserJet, HP DeskJet, and HP DesignJet printers, which are available from Hewlett-Packard Company of Palo Alto, California. (HP, Indigo, Laserjet, DeskJet, and DesignJet are trademarks of Hewlett-Packard Company.)

External client 16 (more generally a RIP control entity) includes a computing device or other entity, such as an application or object, capable of controlling or otherwise communicating with a RIP, such as RIP 126. In the exemplary embodiment, client 16 takes the form of a variable-data-printing

(VDP) manager within or without the same machine or processor(s) that implement digital front end 12. More particularly, external client 16 includes software modules or objects 161, 162, and 163 for communicating respectively with socket ports 1261A, 1261B, and 1261C of controller interface 1261.

Figure 2 shows a flowchart 20 of an exemplary method of operating system 100. Flow chart 20 includes process blocks 202-218, which may generally correspond to machine-readable and executable instructions or sets of instructions stored or carried on a storage medium or carrier signal for implementing various operations and/or components of system 100. Though these blocks are arranged and/or described sequentially, other embodiments may reorder the blocks, omit one or more blocks, combine two or more blocks, and/or execute two or more blocks in parallel using multiple processors or a single processor organized as two or more virtual machines or subprocessors. Moreover, still other embodiments implement the blocks as one or more specific interconnected hardware or integrated-circuit modules with related control and data signals communicated between and through the modules. Thus, this and other exemplary process flows in this document are applicable to software, firmware, hardware, and other types of implementations.

The exemplary method begins at block 202 which entails digital front end 12 (in Figure 1), or more precisely RIP manager 122, receiving at least one print job for processing. In the exemplary embodiment, the print job is received from a device, such as desktop publishing station (not shown), via a socket connection or link to a local-area or wide-area network. The print job, which may take the form of a Job Definition Format (JDF) job ticket, includes vector image data, such as PostScript (PS) data, Portable Document Format (PDF) data, or Printer Control Language (PCL) data, and a rasterization specification. Execution continues at block 204.

In block 204, RIP manager 122 assigns the job (or a portion thereof) to a RIP, such as RIP 126. The assignment, along with the relevant portion of the vector image data from the print job, passes through controller interface 1261 to RIP engine 1262. In the exemplary embodiment, the RIP manager

communicates with RIP 126 (as well as other RIPs) over message channels that reference URLs (Uniform Resource Locators) of files that are transferred asynchronously between the RIP manager and the RIP engine. Execution proceeds to block 206.

Block 206 entails RIP 126 rasterizing the data and transferring the rasterized data to printer set 14 for printing. In the exemplary embodiment, this rasterization occurs in compliance with the rasterization specification using any technology suitable for the given vector image data and the targeted printer. Once rasterization is complete, RIP 126 notifies RIP manager 122 that it has completed its rasterization assignment. In the exemplary embodiment, this notification entails the RIP engine sending a "job done" message to the RIP manager. In response, the RIP manager updates RIP database 1222 to reflect the available status of RIP 126. Execution continues to block 208.

In block 208, RIP manager 122 receives a request from external client 16 to control one of the RIP engines. In the exemplary embodiment, this entails RIP manager 122 receiving a request from external client 16 to use a RIP. However in other embodiments, the request may identify a specific RIP or a set of RIP criteria that RIP manager 122 uses to identify an appropriate RIP, by for example searching RIP database 1222 for available RIPs that are associated with RIP manager 122 and that meet or comply with the RIP criteria. In some embodiments, the client may request a RIP engine specifically for VDP jobs. Execution then continues at block 210.

Block 210 entails RIP manager 112 assigning or transferring control of an appropriate RIP, such as RIP 126, over to the external client. In the exemplary embodiment, this entails the RIP manager sending an "enter direct mode" message via controller interface 1261 (more precisely socket port 1261B) to RIP engine 126, with the message identifying a set of one or more socket ports and a network name of the external client. "Direct mode" refers to a communication mode that allows direct communication between a RIP engine and an external client (or other RIP-control entity) without using the RIP manager as an intermediary or pass-through.

In exemplary response to the "enter direct mode" message, RIP 126 initializes itself for interaction with the client and communicates a "direct mode ready" message to RIP manager 122. Initialization entails launching one or more threads to monitor socket ports 1261A and 126B for incoming data. In turn, RIP manager 112 sends a message identifying the RIP by its IP address to external client 16.

Block 212 entails the external client sending vector image and related control or message data directly to the identified RIP engine, in this case RIP 126. The precise form of this communication and the controller interface for the identified RIP engine is generally contingent on the implementation of external client and the RIP engine. In the exemplary embodiment, the data is sent to one or more input socket ports associated with the controller interface, such as input port 1261A, and ultimately conveyed via socket port 1261D to RIP engine 1262. Other embodiments may omit an intermediate socket port, such as port 1261D.

In block 214, the RIP module rasterizes the received image vector data. In the exemplary embodiment, the RIP engine buffers and rasterizes the data as it is received and outputs the corresponding bit-map data, through the controller interface, more precisely output socket port 1261C, to the external client. When the client has finished sending data to the RIP engine, the client closes the input socket port 1261A, signaling controller interface 1261 for RIP 126 to close intermediate socket port 1261D between the controller interface and the RIP module. The RIP module interprets this closure as an "end of job" signal, and closes the output socket port after all output data has been forwarded to the client (or other location), advancing execution to block 216.

In block 216, the external client notifies the RIP manager that it has finished using the identified or assigned RIP. Execution of the exemplary method then advances to block 218.

Block 218 entails the RIP manager (or other RIP control entity) making the RIP engine available for use by itself and/or other RIP control entities, such as other external clients or other RIP managers. In the exemplary embodiment, this entails the RIP manager sending an "exit direct mode" message to the RIP

engine, moving the virtual representation of the RIP engine to a temporary holding place until the RIP engine reports back to the RIP manager that it is ready for static processing, and updating the status of the RIP engine in the RIP engine database to indicate its availability. In response to the "exit direct mode" message, the RIP engine is restored to a state enabling it to handle requests from the RIP manager. In some instances, the RIP engine may be reassigned to a RIP pipeline to process static jobs.

Conclusion

The embodiments described in this document are intended only to illustrate and teach one or more ways of practicing or implementing the present invention, not to restrict its breadth or scope. The actual scope of the invention, which embraces all ways of practicing or implementing the teachings of the invention, is defined only by the following claims and their equivalents.